МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Черкаський державний технологічний університет

Факультет інформаційних технологій і систем

Кафедра програмного забезпечення автоматизованих систем

**ЗВІТ**

про виконання лабораторної роботи №7

з дисципліни

“Об’єктно-орієнтоване програмування”

Перевірив: Виконав:

Старший викладач студент ПЗС-1944

Заспа Г.О. 2-го курсу ФІТІС

Колісник Іван

Черкаси, 2020

**Тема:** Композиція. Контейнерні класи в Java.

**Мета:** Зрозуміти принцип поліморфізму. Навчитись використовувати поліморфізм при створенні об'єктно-орієнтованих програм.

**Завдання.**

Створити програму, що дає можливість обраховувати домогосподарству витрати на електроенергію. В програмі є декілька класів.

Перший – дані за певний місяць. Він вміщує

**поля:**

назва поточного місяця,

поточний рік,

показник лічильника на кінець попереднього місяця,

показник лічильника на кінець поточного місяця,

кількість використаної електроенергії,

ціна електроенергії в поточному місяці,

вартість електроенергії в поточному місяці;

**методи:**

встановлення значення показника лічильника на кінець попереднього місяця,

встановлення значення показника лічильника на кінець поточного місяця,

обрахунку кількості використаної електроенергії

обрахунку вартості використаної електроенергії

**конструктор:**

що має параметри: місяць, рік;

можливо, інші конструктори.

Також в повинні бути потрібні get та set-методи.

При виклику методів встановлення значення показника лічильника на кінець попереднього місяця або встановлення значення показника лічильника на кінець поточного місяця повинні автоматично проводитись перерахунки кількості використаної електроенергії та вартості використаної електроенергії.

Другий клас – облік електроенергії. Він вміщує колекцію об'єктів даних за певний місяць (об'єктів першого класу), а також:

**поля:**

загальна кількість спожитої електроенергії;

загальна вартість спожитої електроенергії;

**методи:**

обрахунку загальної кількості спожитої електроенергії;

обрахунку загальної вартості спожитої електроенергії;

перевірки коректності даних;

обрахунку кількості та вартості спожитої електроенергії за певний період часу.

Розробити інтерфейс програми, що дає можливість: вводити дані за певний місяць, виводити дані за певний період часу, виводити дані про кількість та вартість спожитої електроенергії за весь період ведення обліку.

**Додатково.** Розробити клас-нащадок першого класу даної програми(дані за певний місяць), який буде заснований на роботі двозонного лічильника, в якому вартість електроенергії, спожитої вночі з 23 до 6 години рахується за тарифом, удвічі нижчим за звичайний.

**Хід роботи**

**Лістинг програми:**

**DataForAllMonth.java**

package com.company;  
import javax.swing.\*;  
import java.util.\*;  
  
public class DataForAllMonths {  
  
 int totalUsedEnergyCost;  
  
 int totalIndexSum;  
  
 Map<Integer, List<ElectricityForMonth>> allData = new TreeMap<>();  
  
 int i;  
  
 void addNewMonthData(ElectricityForMonth data) {  
 List<ElectricityForMonth> thisYearList = allData.get(data.getYear());  
 if (thisYearList != null) {  
 for(int i = 0; i<thisYearList.size(); i++){  
 ElectricityForMonth electricityForMonth = thisYearList.get(i);  
 if(electricityForMonth.getMonth().equals(data.getMonth())){  
 JOptionPane.*showMessageDialog*(null,"Дані за цей місяць вже введено!");  
 return;  
 }  
 }  
 if (data.getPreviousMonthIndex() >= data.getThisMonthIndex()){  
 JOptionPane.*showMessageDialog*(null,"Дані за попередній місяць перевищують данні за поточний!");  
 return;  
 }  
  
 thisYearList.add(data);  
 MonthsComparator monthsComparator = new MonthsComparator();  
 Collections.*sort*(thisYearList, monthsComparator);  
 System.*out*.println(allData.toString());  
 } else {  
 thisYearList = new ArrayList<>();  
 thisYearList.add(data);  
 allData.put(data.getYear(), thisYearList);  
 System.*out*.println(allData.toString());  
 }  
 }  
  
 public List<ElectricityForMonth> getMonthsDataForPeriod(int firstYear, int lastYear, int firstMonthParam, int lastMonthParam) {  
 List<ElectricityForMonth> resultList = new ArrayList<>();  
 for (int i = firstYear; i <= lastYear; i++) {  
 List<ElectricityForMonth> oneYearElectroData = allData.get(i);  
 if (oneYearElectroData != null) {  
 if (i == firstYear) {  
 int foundIndex = getMonthIndex(firstMonthParam, oneYearElectroData);  
 resultList.addAll(oneYearElectroData.subList(foundIndex, oneYearElectroData.size()));  
 } else {  
 if (i == lastYear) {  
 int foundIndex = getMonthIndex(lastMonthParam, oneYearElectroData);  
 resultList.addAll(oneYearElectroData.subList(0, foundIndex));  
 } else {  
 resultList.addAll(oneYearElectroData);  
 }  
 }  
 }  
 }  
 return resultList;  
 }  
  
 private int getMonthIndex(int monthNumber, List<ElectricityForMonth> oneYearElectroData) {  
 int foundIndex = 0;  
 for (int j = 0; j < oneYearElectroData.size(); j++) {  
 ElectricityForMonth electricityForMonth = oneYearElectroData.get(j);  
 if (electricityForMonth.getMonth().ordinal() == monthNumber) {  
 foundIndex = j;  
 break;  
 }  
 }  
 return foundIndex;  
 }  
  
 public void calculateTotalSumAndCost(){  
  
 for(Map.Entry<Integer, List<ElectricityForMonth>> entry : allData.entrySet()){  
 List<ElectricityForMonth> electricityForMonths = entry.getValue();  
 for (ElectricityForMonth electricityForMonth : electricityForMonths) {  
 totalIndexSum += (electricityForMonth.getThisMonthIndex()-electricityForMonth.getPreviousMonthIndex());  
 int costForPeriod = (electricityForMonth.getThisMonthIndex()-electricityForMonth.getPreviousMonthIndex())\*electricityForMonth.getCurrentPrice();  
 totalUsedEnergyCost += costForPeriod;  
 }  
 }  
  
 }  
}

**ElectricityForMonth.java:**

package com.company;  
public class ElectricityForMonth {  
  
 Months monthName;  
 int year;  
 private int previousMonthIndex;  
 private int thisMonthIndex;  
 int currentPrice;  
  
 ElectricityForMonth(Months monthName, int year){  
 this.monthName=monthName;  
 this.year=year;  
 }  
  
 ElectricityForMonth(Months monthName, int year, int lastIndex, int currentIndex, int currentPrice){  
 this.monthName=monthName;  
 this.year=year;  
 this.previousMonthIndex =lastIndex;  
 this.thisMonthIndex =currentIndex;  
 this.currentPrice=currentPrice;  
 }  
  
 String getMonthData(){  
 return "Місяць:"+ monthName+ ";" + " Рік:" + year +";"+ " Поп.місяць:" + previousMonthIndex +";"+ " Пот. місяць:" + thisMonthIndex + ";" + " Ціна :" + currentPrice + ";" ;  
 }  
  
 @Override  
 public String toString(){  
 return "Місяць:"+ monthName+ ";" + " Рік:" + year +";"+ " Поп. місяць:" + previousMonthIndex +";"+ " Пот. місяць:" + thisMonthIndex + ";" + " Ціна :" + currentPrice + ";" ;  
 }  
 public Months getMonth() {  
 return monthName;  
 }  
  
 public void setMonthName(Months monthName) {  
 this.monthName = monthName;  
 }  
  
 public int getYear() {  
 return year;  
 }  
  
 public void setYear(int year) {  
 this.year = year;  
 }  
  
 public int getPreviousMonthIndex() {  
 return previousMonthIndex;  
 }  
  
 public void setPreviousMonthIndex(int previousMonthIndex) {  
 this.previousMonthIndex = previousMonthIndex;  
 }  
  
 public int getThisMonthIndex() {  
 return thisMonthIndex;  
 }  
  
 public void setThisMonthIndex(int thisMonthIndex) {  
 this.thisMonthIndex = thisMonthIndex;  
 }  
  
 public int getCurrentPrice() {  
 return currentPrice;  
 }  
  
 public void setCurrentPrice(int currentPrice) {  
 this.currentPrice = currentPrice;  
 }  
}

**Months.java**

package com.company;  
  
public enum Months {  
 *JANUARY*("January"),*FEBRUARY*("February"),*MARCH*("March"),  
 *APRIL*("April"),*MAY*("May"),*JUNE*("June"),  
 *JULY*("July"),*AUGUST*("August"),*SEPTEMBER*("September"),  
 *OCTOBER*("October"),*NOVEMBER*("November"),*DECEMBER*("December");  
  
 String monthName;  
  
 public String getMonthName() {  
 return monthName;  
 }  
  
 Months(String monthName){  
 this.monthName = monthName;  
 }  
  
}

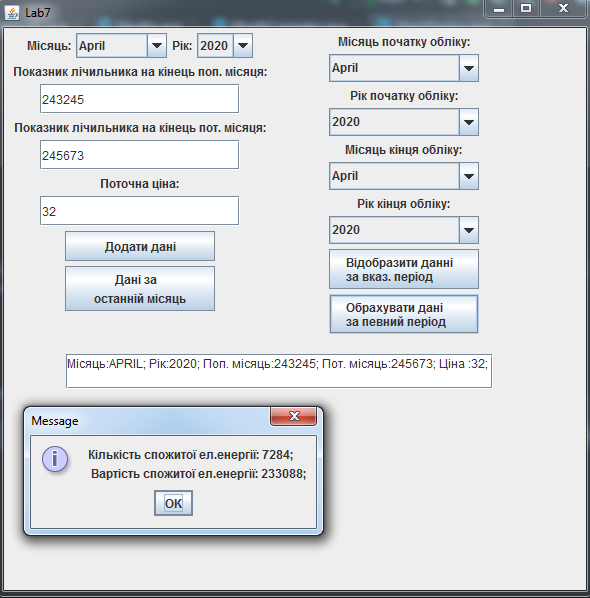
**MonthCompator.java:**

package com.company;  
import java.util.Comparator;  
  
class MonthsComparator implements Comparator<ElectricityForMonth> {  
  
 public int compare (ElectricityForMonth m1, ElectricityForMonth m2){  
 if(m1.monthName == m2.monthName){  
 return 0;  
 }  
 if(m1.monthName.ordinal() > m2.monthName.ordinal()) {  
 return 1;  
 }  
 else{  
 return -1;  
 }  
 }  
}

**MainWindow.java:**

package com.company;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class MainWindow {  
  
 private ElectricityForMonth firstMonth;  
 private DataForAllMonths dataForAllMonths;  
 private Months months;  
  
 private JFrame frame;  
  
 private JPanel panel;  
 private JPanel infoPanel;  
 private JPanel counterPanel;  
  
 private JButton addData;  
 private JButton showDataForCurrentMonth;  
 private JButton countButton;  
 private JButton showDataForPeriod;  
  
 private JTextField lastIndexField;  
 private JTextField currentIndexField;  
 private JTextField currentPriceField;  
  
 private JTextArea infoForPeriod = new JTextArea();  
  
 private JComboBox monthBox;  
 private JComboBox yearBox;  
  
 private JComboBox startingMonthBox;  
 private JComboBox finalMonthBox;  
 private JComboBox startingYearBox;  
 private JComboBox finalYearBox;  
  
 private int chooseMonth;  
 private int chooseYear;  
  
 private JLabel description;  
 private JLabel sumForPeriodLabel;  
  
 String allData = new String();  
 String sumForPeriod;  
  
 String[] monthList = {"January","February","March","April","May","June","July","August","September","October","November","December"};  
  
 String[] yearList = {"2019","2020"};  
  
 Months month;  
 int monthIndex;  
  
 int year;  
  
 int m1;  
 int m2;  
  
 int startingYear;  
 int finalYear ;  
  
  
 MainWindow() {  
 dataForAllMonths = new DataForAllMonths();  
 this.frame = new JFrame("Lab7");  
 this.buildPanelForAdding();  
 this.panelForCounting();  
 }  
  
 private void buildPanelForAdding() {  
 this.panel = new JPanel();  
 this.panel.setLayout(new FlowLayout());  
  
 this.panel.add(new JLabel("Місяць:"));  
  
 this.monthBox = new JComboBox(monthList);  
 monthBox.setEditable(false);  
 this.panel.add(this.monthBox);  
  
 this.panel.add(new JLabel("Рік:"));  
  
 this.yearBox = new JComboBox(yearList);  
 yearBox.setEditable(false);  
 this.panel.add(this.yearBox);  
  
 this.panel.add(new JLabel("Показник лічильника на кінець поп. місяця:"));  
  
 this.lastIndexField = new JTextField();  
 this.lastIndexField.setPreferredSize(new Dimension(200, 30));  
 this.panel.add(this.lastIndexField);  
  
 this.panel.add(new JLabel("Показник лічильника на кінець пот. місяця:"));  
  
 this.currentIndexField = new JTextField();  
 this.currentIndexField.setPreferredSize(new Dimension(200, 30));  
 this.panel.add(this.currentIndexField);  
  
 this.panel.add(new JLabel("Поточна ціна:"));  
  
 this.currentPriceField = new JTextField();  
 this.currentPriceField.setPreferredSize(new Dimension(200, 30));  
 this.panel.add(this.currentPriceField);  
  
 this.addData = new JButton("Додати дані");  
 this.addData.setPreferredSize(new Dimension(150, 30));  
 this.panel.add(this.addData);  
  
 this.showDataForCurrentMonth = new JButton();  
 this.showDataForCurrentMonth.setLayout(new BorderLayout());  
  
 this.showDataForCurrentMonth.setPreferredSize(new Dimension(150, 45));  
 this.panel.add(this.showDataForCurrentMonth);  
 JLabel label1 = new JLabel(" Дані за ");  
 JLabel label2 = new JLabel(" останній місяць");  
  
 showDataForCurrentMonth.add(BorderLayout.*NORTH*,label1);  
 showDataForCurrentMonth.add(BorderLayout.*SOUTH*,label2);  
 this.panel.add(this.showDataForCurrentMonth);  
  
 this.addData.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 chooseMonth = monthBox.getSelectedIndex();  
 chooseYear = yearBox.getSelectedIndex();  
 inputNewMonthData();  
 }  
 });  
  
 this.showDataForCurrentMonth.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 showInfo();  
 }  
 });  
  
 this.frame.setSize(550, 600);  
 this.frame.setLayout(null);  
 this.frame.add(this.panel);  
 this.panel.setBounds(1, 0, 270, 320);  
 this.frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 this.frame.setVisible(true);  
 }  
  
 private void inputNewMonthData(){  
  
 monthIndex = this.monthBox.getSelectedIndex();  
 month = Months.*values*()[monthIndex];  
 year = Integer.*parseInt*((String)this.yearBox.getSelectedItem());  
  
 this.firstMonth = new ElectricityForMonth(month,year,Integer.*valueOf*(this.lastIndexField.getText()),Integer.*valueOf*(this.currentIndexField.getText()),Integer.*valueOf*(this.currentPriceField.getText()));  
 this.dataForAllMonths.addNewMonthData(firstMonth);  
  
 infoForPeriod.setText("");  
  
 dataForAllMonths.totalIndexSum = 0;  
 dataForAllMonths.totalUsedEnergyCost = 0;  
 sumForPeriod = "";  
  
 }  
  
 private void showInfo(){  
  
 String info;  
  
 this.infoPanel = new JPanel();  
 this.infoPanel.setLayout(null);  
  
 info=firstMonth.getMonthData();  
 this.firstMonth = new ElectricityForMonth(null,0,0,0,0);  
  
 JOptionPane.*showMessageDialog*(null,""+ info);  
  
 // System.out.println(firstMonth.getMonthData());  
//  
// this.description = new JLabel();  
// this.description.setText(info);  
// this.infoPanel.add(description);  
//  
// this.frame.add(infoPanel);  
// this.infoPanel.setBounds(1,522 ,550,230);  
// this.infoPanel.setVisible(true);  
//  
// this.frame.getContentPane().setVisible(false);  
// this.frame.getContentPane().setVisible(true);  
  
 }  
 private void showDataForPeriod(){  
  
 m1 = this.startingMonthBox.getSelectedIndex();  
 m2 = this.finalMonthBox.getSelectedIndex();  
  
 startingYear = Integer.*parseInt*((String)this.startingYearBox.getSelectedItem());  
 finalYear = Integer.*parseInt*((String)this.finalYearBox.getSelectedItem());  
  
 if(m1 > m2){  
 JOptionPane.*showMessageDialog*(null,"Некоректинй порядок введення місяців!");  
 return;  
 }  
 if (startingYear>finalYear){  
 JOptionPane.*showMessageDialog*(null,"Некоректинй порядок введення років!");  
 return;  
 }  
 List<ElectricityForMonth> monthsInfo = dataForAllMonths.getMonthsDataForPeriod(startingYear,finalYear,m1,m2);  
  
 for (int i = 0; i < monthsInfo.size(); i++) {  
 allData += String.*valueOf*(monthsInfo.get(i))+ " \n";  
 }  
 System.*out*.println(monthsInfo.toString());  
  
 this.infoPanel = new JPanel();  
  
 this.infoForPeriod.setVisible(true);  
 this.infoForPeriod.setSize(450,100);  
 this.infoForPeriod.setText(allData);  
 this.infoForPeriod.setEditable(false);  
  
 JScrollPane scroll = new JScrollPane(infoForPeriod, ScrollPaneConstants.*VERTICAL\_SCROLLBAR\_AS\_NEEDED*,  
 ScrollPaneConstants.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);  
  
 this.frame.add(infoPanel);  
 this.infoPanel.setBounds(1,321 ,550,230);  
 this.infoPanel.setVisible(true);  
 this.infoPanel.add(scroll);  
  
 this.frame.getContentPane().setVisible(false);  
 this.frame.getContentPane().setVisible(true);  
  
 allData ="";  
 }  
  
 private void countDataForPeriod(){  
  
 dataForAllMonths.calculateTotalSumAndCost();  
  
 sumForPeriod = "Кількість спожитої ел.енергії: "+ dataForAllMonths.totalIndexSum + "; \n" +  
 " Вартість спожитої ел.енергії: "+ dataForAllMonths.totalUsedEnergyCost +";";  
  
 JOptionPane.*showMessageDialog*(null,sumForPeriod);  
// this.sumForPeriodLabel = new JLabel();  
// this.sumForPeriodLabel.setText(sumForPeriod);  
// this.sumForPeriodLabel.setBounds(1,422,550,110);  
// this.infoPanel.add(sumForPeriodLabel);  
//  
// this.frame.getContentPane().setVisible(false);  
// this.frame.getContentPane().setVisible(true);  
  
 }  
  
  
 private void panelForCounting() {  
  
 this.counterPanel = new JPanel();  
  
 this.counterPanel.add(new JLabel("Місяць початку обліку:"));  
  
 this.startingMonthBox = new JComboBox(monthList);  
 startingMonthBox.setEditable(false);  
 this.startingMonthBox.setPreferredSize(new Dimension(150, 28));  
 this.counterPanel.add(this.startingMonthBox);  
  
 this.counterPanel.add(new JLabel("Рік початку обліку:"));  
  
 this.startingYearBox = new JComboBox(yearList);  
 startingYearBox.setEditable(false);  
 this.startingYearBox.setPreferredSize(new Dimension(150, 28));  
 this.counterPanel.add(this.startingYearBox);  
  
 this.counterPanel.add(new JLabel("Місяць кінця обліку:"));  
  
 this.finalMonthBox = new JComboBox(monthList);  
 finalMonthBox.setEditable(false);  
 this.finalMonthBox.setPreferredSize(new Dimension(150, 28));  
 this.counterPanel.add(this.finalMonthBox);  
  
 this.counterPanel.add(new JLabel("Рік кінця обліку:"));  
  
 this.finalYearBox = new JComboBox(yearList);  
 finalYearBox.setEditable(false);  
 this.finalYearBox.setPreferredSize(new Dimension(150, 28));  
 this.counterPanel.add(this.finalYearBox);  
  
 this.showDataForPeriod = new JButton();  
 this.showDataForPeriod.setLayout(new BorderLayout());  
  
 this.showDataForPeriod.setPreferredSize(new Dimension(150, 40));  
 this.counterPanel.add(this.showDataForPeriod);  
 JLabel label1 = new JLabel("Відобразити данні");  
 JLabel label2 = new JLabel("за вказ. період");  
  
 showDataForPeriod.add(BorderLayout.*NORTH*,label1);  
 showDataForPeriod.add(BorderLayout.*SOUTH*,label2);  
  
 this.countButton = new JButton();  
 this.countButton.setLayout(new BorderLayout());  
  
 this.countButton.setPreferredSize(new Dimension(150, 40));  
 this.counterPanel.add(this.countButton);  
 JLabel label3 = new JLabel("Обрахувати дані");  
 JLabel label4 = new JLabel("за певний період");  
  
 countButton.add(BorderLayout.*NORTH*,label3);  
 countButton.add(BorderLayout.*SOUTH*,label4);  
 this.counterPanel.add(this.countButton);  
  
  
 this.showDataForPeriod.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 showDataForPeriod();  
 }  
 });  
  
 this.countButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 countDataForPeriod();  
 }  
 });  
  
 this.frame.add(this.counterPanel);  
 this.counterPanel.setBounds(300 ,0, 200 ,320);  
  
  
 }  
}

**Результат виконання програми:**





**Висновок:** Я зрозуміти принципи та призначення композиції та навчитись використовувати її при побудові класів. Навчитись використовувати контейнерні класи.